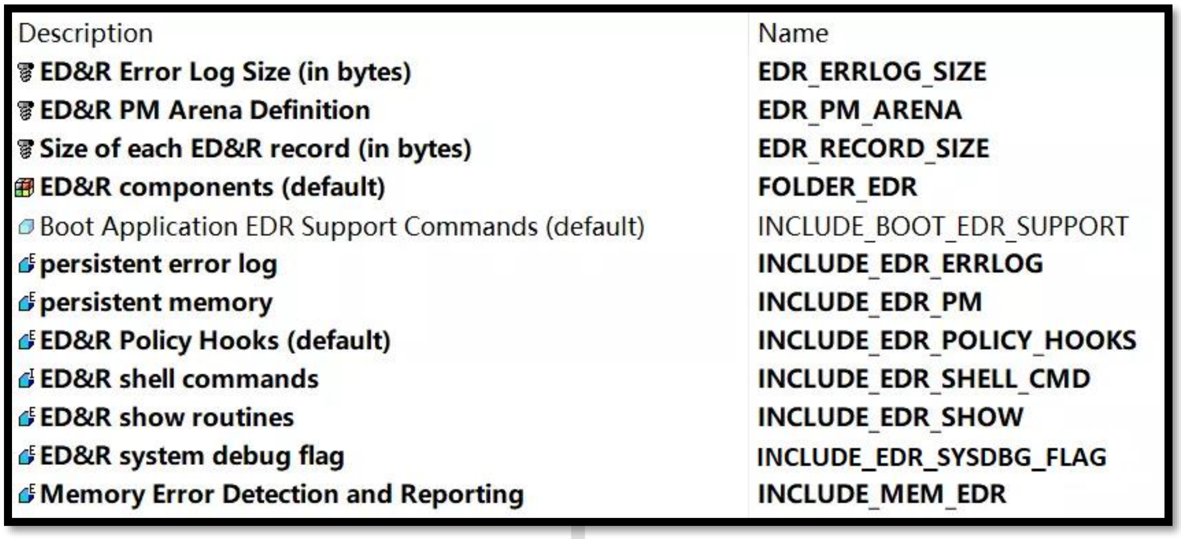
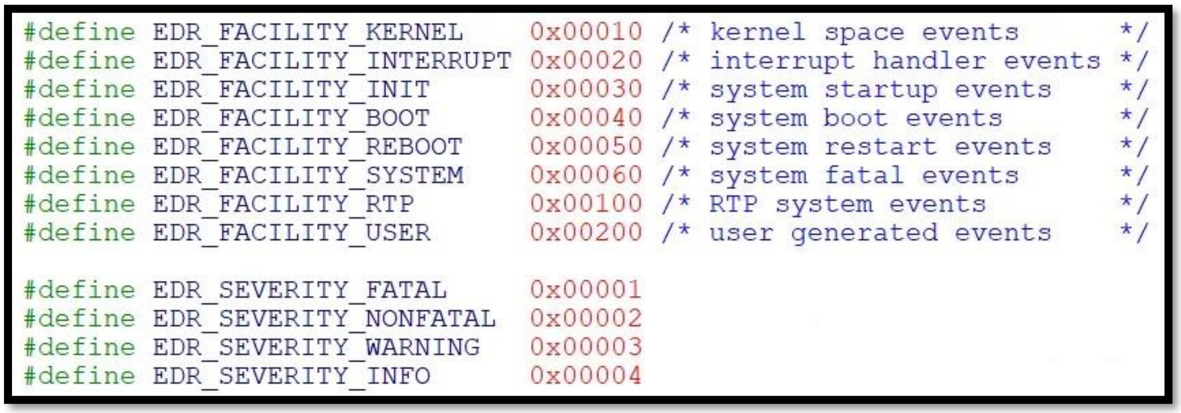
VxWorks Component 之 EDR

EDR，Error Detection and Reporting，Vx6 新加的一种调试机制，可以监测、记录系统错误



Description	Name
ED&R Error Log Size (in bytes)	EDR_ERRLOG_SIZE
ED&R PM Arena Definition	EDR_PM_ARENA
Size of each ED&R record (in bytes)	EDR_RECORD_SIZE
ED&R components (default)	FOLDER_EDR
Boot Application EDR Support Commands (default)	INCLUDE_BOOT_EDR_SUPPORT
persistent error log	INCLUDE_EDR_ERRLOG
persistent memory	INCLUDE_EDR_PM
ED&R Policy Hooks (default)	INCLUDE_EDR_POLICY_HOOKS
ED&R shell commands	INCLUDE_EDR_SHELL_CMD
ED&R show routines	INCLUDE_EDR_SHOW
ED&R system debug flag	INCLUDE_EDR_SYSDBG_FLAG
Memory Error Detection and Reporting	INCLUDE_MEM_EDR

包含这个组件后，VxWorks 就会在内存的最高位置分配一块区域，用于循环记录一些系统日志。而这块区域在系统热重启时，并不会清零。记录的这些日志，可以分为多种事件：内核、中断、进程、等等；还可以分为不同的严重程度：致命或警告等

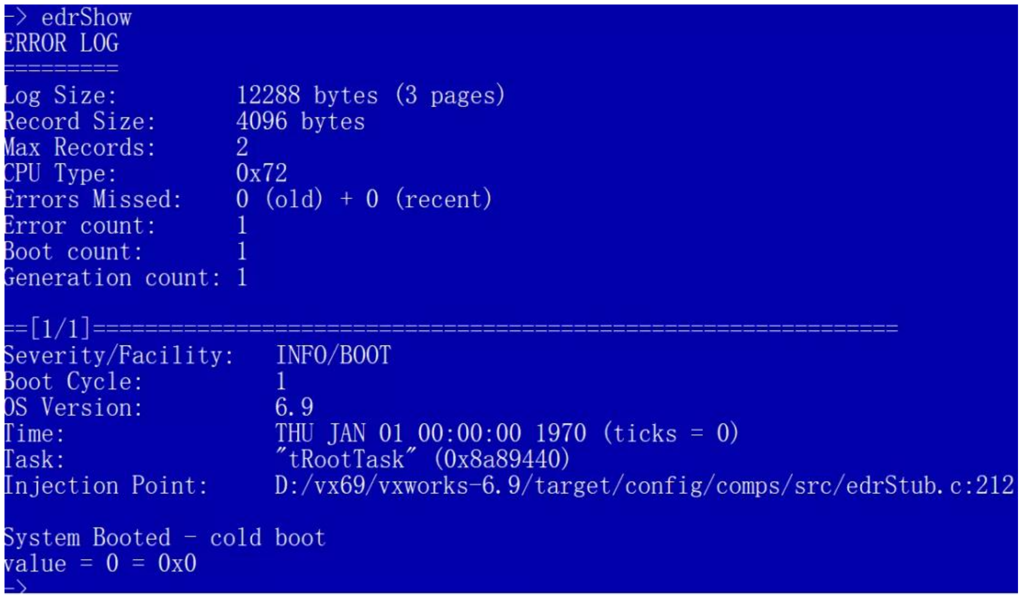


```
#define EDR_FACILITY_KERNEL    0x00010 /* kernel space events    */
#define EDR_FACILITY_INTERRUPT 0x00020 /* interrupt handler events */
#define EDR_FACILITY_INIT      0x00030 /* system startup events  */
#define EDR_FACILITY_BOOT      0x00040 /* system boot events     */
#define EDR_FACILITY_REBOOT    0x00050 /* system restart events  */
#define EDR_FACILITY_SYSTEM    0x00060 /* system fatal events    */
#define EDR_FACILITY_RTP       0x00100 /* RTP system events      */
#define EDR_FACILITY_USER      0x00200 /* user generated events  */

#define EDR_SEVERITY_FATAL     0x00001
#define EDR_SEVERITY_NONFATAL  0x00002
#define EDR_SEVERITY_WARNING   0x00003
#define EDR_SEVERITY_INFO      0x00004
```

具体到每条日志，包含了时间、事件类型和严重程度、OS 版本、任务信息、进程信息、日志的创建位置等等。如果系统还包含其它一些相应组件，日志里还会包含更多的信息

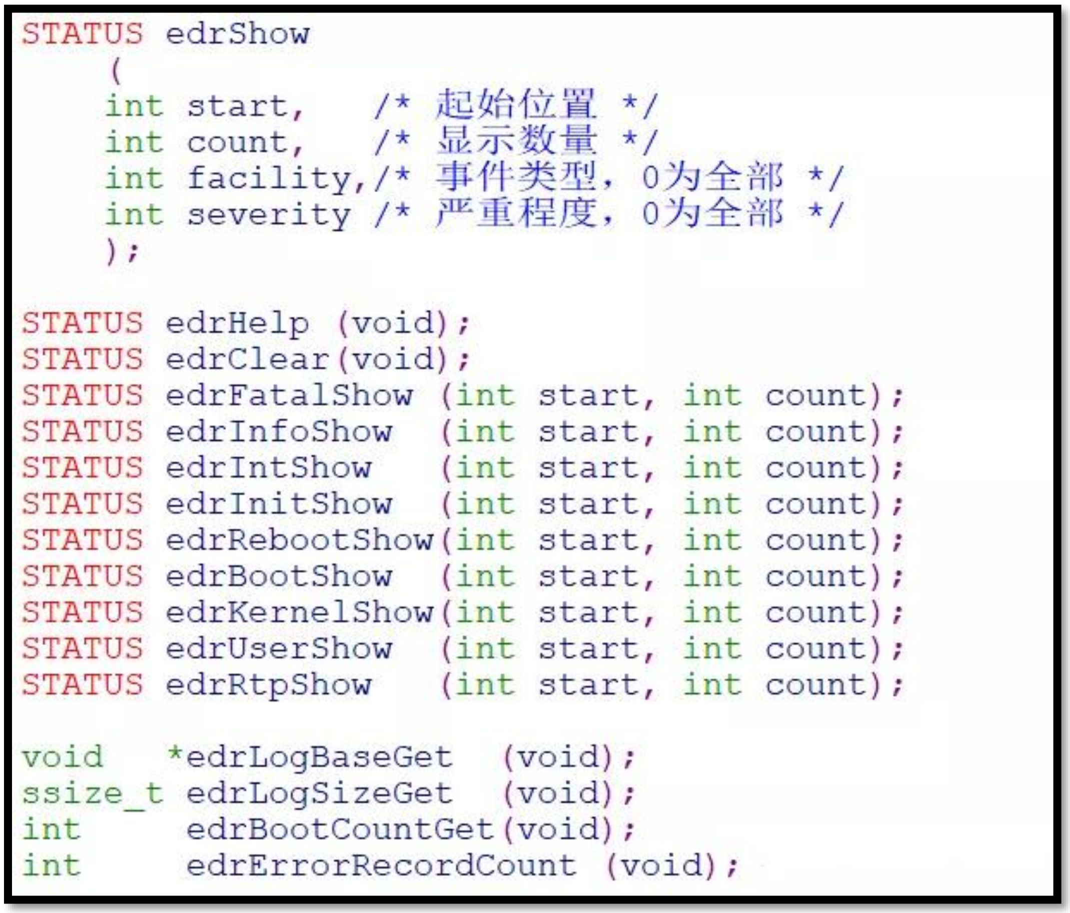
- INCLUDE_EXC_SHOW – 异常信息
- INCLUDE_TASK_SHOW – 寄存器信息
- INCLUDE_DEBUG – 错误地址附件的汇编指令
- INCLUDE_DEBUG – 调用栈的信息

要想查看这些日志，不建议直接查看 EDR 的地址，而是使用函数 edrShow()

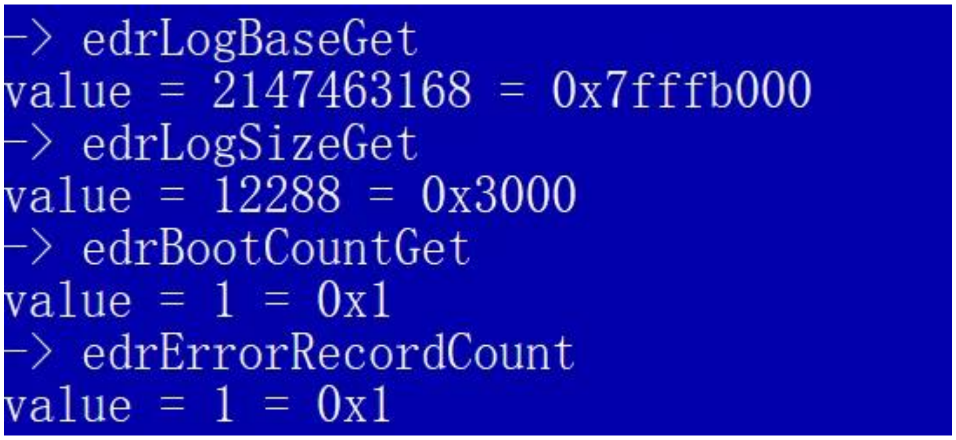


```
-> edrShow
ERROR LOG
=========
Log Size:         12288 bytes (3 pages)
Record Size:      4096 bytes
Max Records:      2
CPU Type:         0x72
Errors Missed:    0 (old) + 0 (recent)
Error count:      1
Boot count:       1
Generation count: 1

==[1/1]============================================================
Severity/Facility:   INFO/BOOT
Boot Cycle:          1
OS Version:          6.9
Time:                THU JAN 01 00:00:00 1970 (ticks = 0)
Task:                "tRootTask" (0x8a89440)
Injection Point:     D:/vx69/vxworks-6.9/target/config/comps/src/edrStub.c:212

System Booted - cold boot
value = 0 = 0x0
->
```

其它几个函数不太常用，有兴趣可以试试。edrClear()用于清除这些日志，否则当存储区域满了以后，新日志**自动**覆盖最老的日志

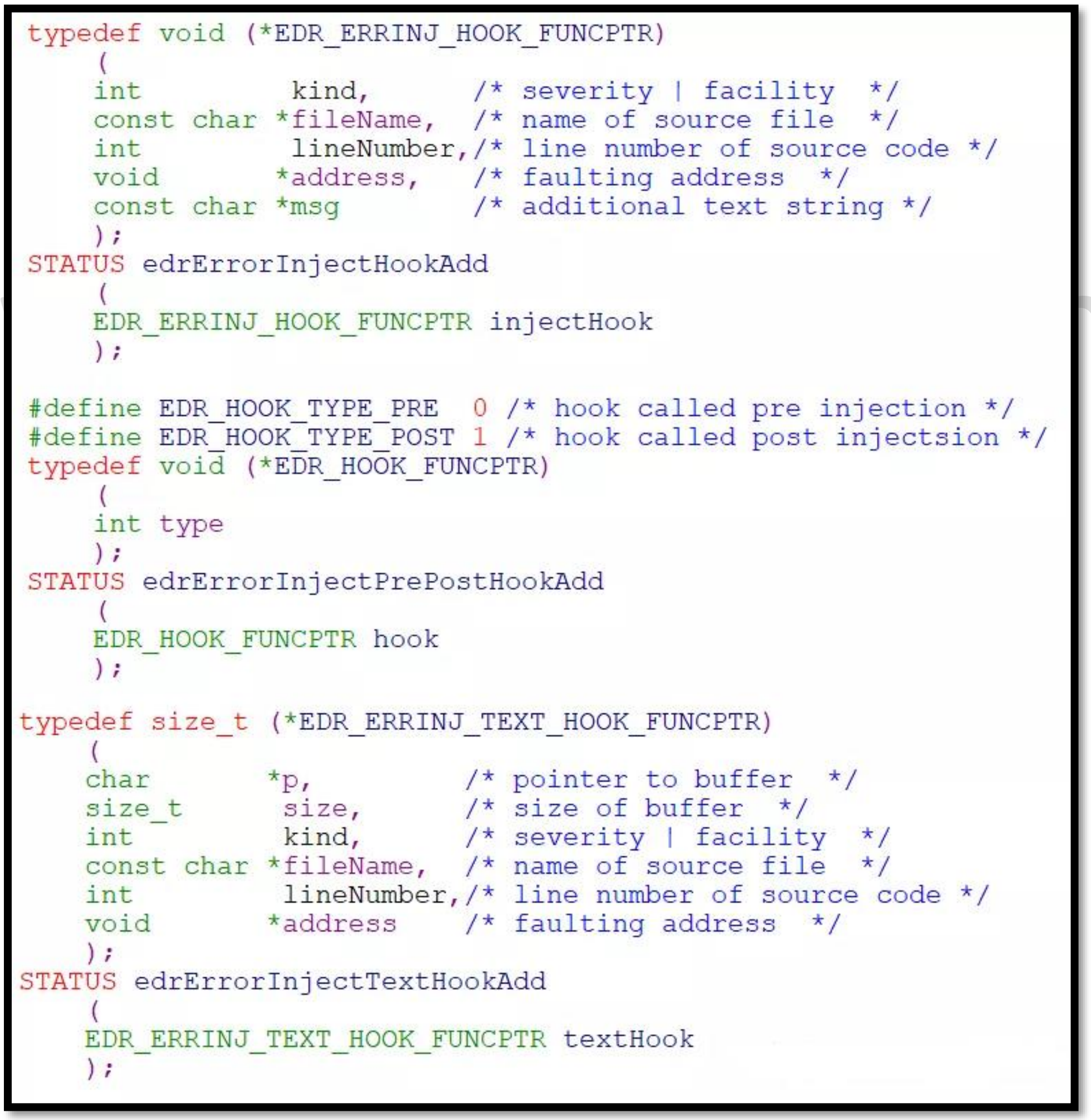


```
STATUS edrShow
    (
    int start,    /* 起始位置 */
    int count,    /* 显示数量 */
    int facility,/* 事件类型, 0为全部 */
    int severity /* 严重程度, 0为全部 */
    );

STATUS edrHelp (void);
STATUS edrClear(void);
STATUS edrFatalShow (int start, int count);
STATUS edrInfoShow  (int start, int count);
STATUS edrIntShow   (int start, int count);
STATUS edrInitShow  (int start, int count);
STATUS edrRebootShow(int start, int count);
STATUS edrBootShow  (int start, int count);
STATUS edrKernelShow(int start, int count);
STATUS edrUserShow  (int start, int count);
STATUS edrRtpShow   (int start, int count);

void    *edrLogBaseGet  (void);
ssize_t edrLogSizeGet   (void);
int     edrBootCountGet(void);
int     edrErrorRecordCount (void);
```



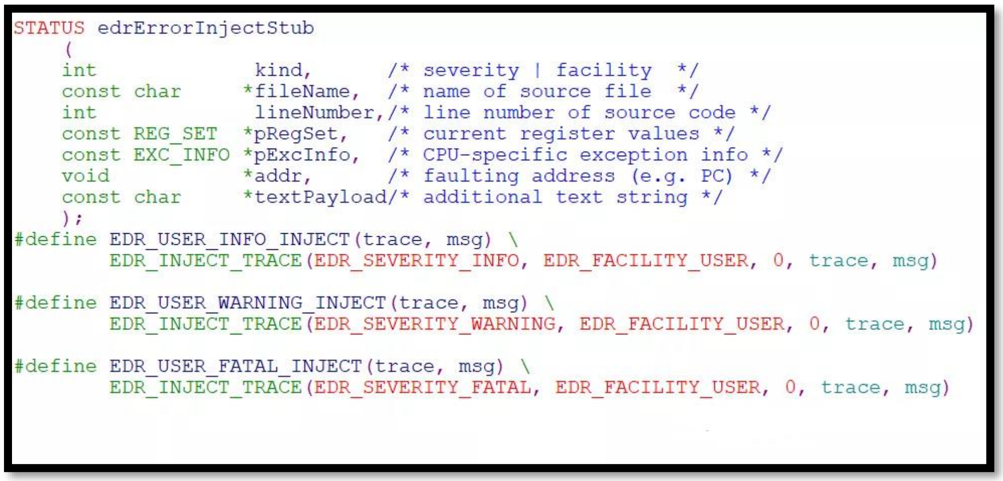
```
-> edrLogBaseGet
value = 2147463168 = 0x7fffb000
-> edrLogSizeGet
value = 12288 = 0x3000
-> edrBootCountGet
value = 1 = 0x1
-> edrErrorRecordCount
value = 1 = 0x1
```

使用者还可以在 EDR 中添加一些钩子函数，当 EDR 生成日志会自动调用它们



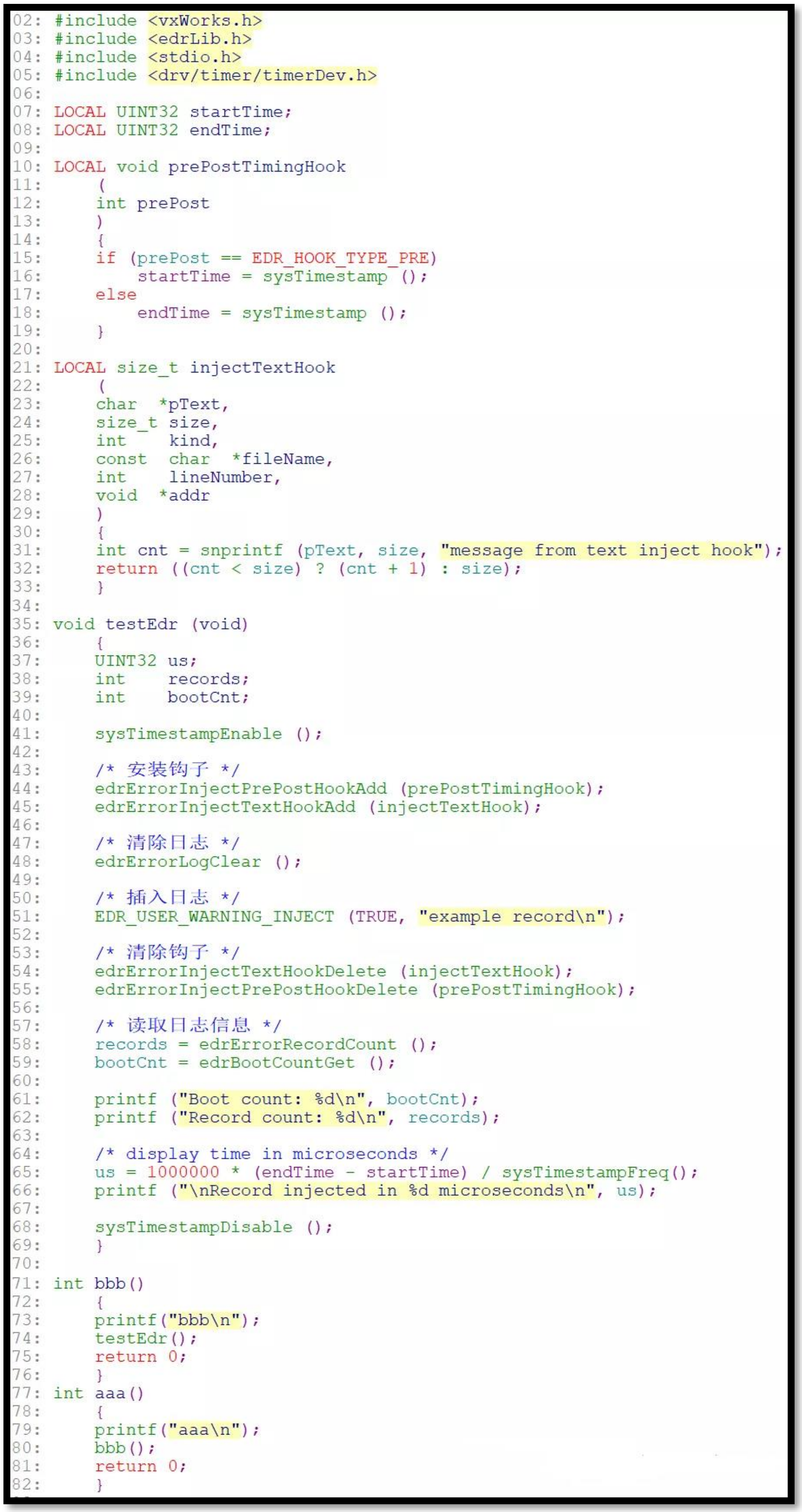
```
typedef void (*EDR_ERRINJ_HOOK_FUNCPTR)
    (
    int          kind,       /* severity | facility  */
    const char *fileName,    /* name of source file  */
    int          lineNumber,/* line number of source code */
    void        *address,    /* faulting address   */
    const char *msg          /* additional text string */
    );
STATUS edrErrorInjectHookAdd
    (
    EDR_ERRINJ_HOOK_FUNCPTR injectHook
    );

#define EDR_HOOK_TYPE_PRE  0 /* hook called pre injection */
#define EDR_HOOK_TYPE_POST 1 /* hook called post injectsion */
typedef void (*EDR_HOOK_FUNCPTR)
    (
    int type
    );
STATUS edrErrorInjectPrePostHookAdd
    (
    EDR_HOOK_FUNCPTR hook
    );

typedef size_t (*EDR_ERRINJ_TEXT_HOOK_FUNCPTR)
    (
    char        *p,          /* pointer to buffer  */
    size_t       size,       /* size of buffer  */
    int          kind,       /* severity | facility  */
    const char *fileName,    /* name of source file  */
    int          lineNumber,/* line number of source code */
    void        *address     /* faulting address   */
    );
STATUS edrErrorInjectTextHookAdd
    (
    EDR_ERRINJ_TEXT_HOOK_FUNCPTR textHook
    );
```

使用者也可以手动生成日志



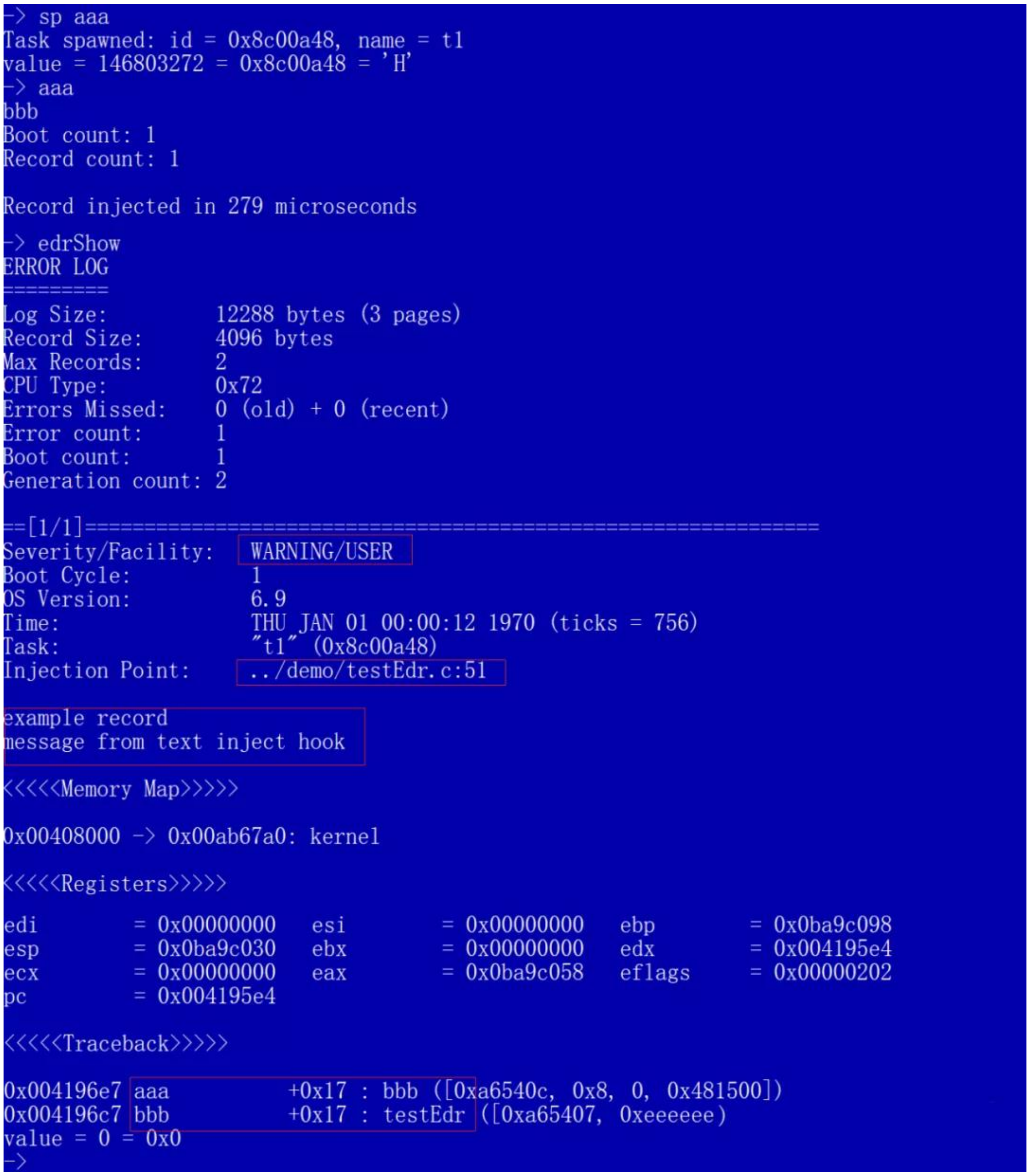
```
STATUS edrErrorInjectStub
    (
    int             kind,       /* severity | facility  */
    const char     *fileName,   /* name of source file  */
    int             lineNumber,/* line number of source code */
    const REG_SET  *pRegSet,    /* current register values */
    const EXC_INFO *pExcInfo,   /* CPU-specific exception info */
    void           *addr,       /* faulting address (e.g. PC) */
    const char     *textPayload/* additional text string */
    );
#define EDR_USER_INFO_INJECT(trace, msg) \
        EDR_INJECT_TRACE(EDR_SEVERITY_INFO, EDR_FACILITY_USER, 0, trace, msg)

#define EDR_USER_WARNING_INJECT(trace, msg) \
        EDR_INJECT_TRACE(EDR_SEVERITY_WARNING, EDR_FACILITY_USER, 0, trace, msg)

#define EDR_USER_FATAL_INJECT(trace, msg) \
        EDR_INJECT_TRACE(EDR_SEVERITY_FATAL, EDR_FACILITY_USER, 0, trace, msg)
```

写个例子看看效果



```c
#include <vxWorks.h>
#include <edrLib.h>
#include <stdio.h>
#include <drv/timer/timerDev.h>

LOCAL UINT32 startTime;
LOCAL UINT32 endTime;

LOCAL void prePostTimingHook
    (
    int prePost
    )
    {
    if (prePost == EDR_HOOK_TYPE_PRE)
        startTime = sysTimestamp ();
    else
        endTime = sysTimestamp ();
    }

LOCAL size_t injectTextHook
    (
    char   *pText,
    size_t size,
    int     kind,
    const  char  *fileName,
    int     lineNumber,
    void   *addr
    )
    {
    int cnt = snprintf (pText, size, "message from text inject hook");
    return ((cnt < size) ? (cnt + 1) : size);
    }

void testEdr (void)
    {
    UINT32 us;
    int     records;
    int     bootCnt;

    sysTimestampEnable ();

    /* 安装钩子 */
    edrErrorInjectPrePostHookAdd (prePostTimingHook);
    edrErrorInjectTextHookAdd (injectTextHook);

    /* 清除日志 */
    edrErrorLogClear ();

    /* 插入日志 */
    EDR_USER_WARNING_INJECT (TRUE, "example record\n");

    /* 清除钩子 */
    edrErrorInjectTextHookDelete (injectTextHook);
    edrErrorInjectPrePostHookDelete (prePostTimingHook);

    /* 读取日志信息 */
    records = edrErrorRecordCount ();
    bootCnt = edrBootCountGet ();

    printf ("Boot count: %d\n", bootCnt);
    printf ("Record count: %d\n", records);

    /* display time in microseconds */
    us = 1000000 * (endTime - startTime) / sysTimestampFreq();
    printf ("\nRecord injected in %d microseconds\n", us);

    sysTimestampDisable ();
    }

int bbb()
    {
    printf("bbb\n");
    testEdr();
    return 0;
    }
int aaa()
    {
    printf("aaa\n");
    bbb();
    return 0;
    }
```

使用 edrShow()可以看到 EDR_USER_WARNING_INJECT()插入的日志，以及任务的函数调用栈



```
-> sp aaa
Task spawned: id = 0x8c00a48, name = t1
value = 146803272 = 0x8c00a48 = 'H'
-> aaa
bbb
Boot count: 1
Record count: 1

Record injected in 279 microseconds
-> edrShow
ERROR LOG
=========
Log Size:          12288 bytes (3 pages)
Record Size:       4096 bytes
Max Records:       2
CPU Type:          0x72
Errors Missed:     0 (old) + 0 (recent)
Error count:       1
Boot count:        1
Generation count:  2

==[1/1]==============================================================
Severity/Facility:    WARNING/USER
Boot Cycle:           1
OS Version:           6.9
Time:                 THU JAN 01 00:00:12 1970 (ticks = 756)
Task:                 "t1" (0x8c00a48)
Injection Point:      ../demo/testEdr.c:51

example record
message from text inject hook

<<<<<Memory Map>>>>>

0x00408000 -> 0x00ab67a0: kernel

<<<<<Registers>>>>>

edi        = 0x00000000   esi        = 0x00000000   ebp        = 0x0ba9c098
esp        = 0x0ba9c030   ebx        = 0x00000000   edx        = 0x004195e4
ecx        = 0x00000000   eax        = 0x0ba9c058   eflags     = 0x00000202
pc         = 0x004195e4

<<<<<Traceback>>>>>

0x004196e7 aaa          +0x17 : bbb ([0xa6540c, 0x8, 0, 0x481500])
0x004196c7 bbb          +0x17 : testEdr ([0xa65407, 0xeeeeeee)
value = 0 = 0x0
->
```